DE GRUYTER OPEN

JAISCR, 2013, Vol. 3, No. 3, pp. 215 – 223
DOI 10.2478/jaiscr-2014-0015

# PARALLEL PBIL APPLIED TO POWER SYSTEM CONTROLLER DESIGN

Komla Folly
*Department of Electrical Engineering, University of Cape Town Private Bag, Rondebosch 7701, South Africa*

**Abstract**

Population-Based Incremental Learning (PBIL) algorithm is a combination of evolutionary optimization and competitive learning derived from artificial neural networks. PBIL has recently received increasing attention in various engineering fields due to its effectiveness, easy implementation and robustness. Despite these strengths, it was reported in the last few years that PBIL suffers from issues of loss of diversity in the population. To deal with this shortcoming, this paper uses parallel PBIL based on multi-population. In parallel PBIL, two populations are used where both probability vectors (PVs) are initialized to 0.5. It is believed that by introducing two populations, the diversity in the population can be increased and better results can be obtained. The approach is applied to power system controller design. Simulations results show that the parallel PBIL approach performs better than the standard PBIL and is as effective as another diversity increasing PBIL called adaptive PBIL.

## 1 Introduction

Population-Based Incremental Learning is a relatively new type of Evolutionary Algorithm (EA) that combines aspects of Genetic Algorithms (GAs) and simple competitive learning derived from Artificial Neural Networks (ANNs). It was first proposed by Baluja [1] in response of the growing need to have a simpler and yet robust Evolutionary Algorithms which is easy to use by the larger community of users who are not necessarily experts in EAs [2], [3].

Like other EAs such as GAs [4]-[6], Differential Evolution (DE) [7], [8], and variants such as Ant Colony optimization (ACO) [9], and Particle Swarm Optimization (PSO) [10]-[11], PBIL works with a population of individuals rather than a single individual (e.g., point) [12]-[16]. Over successive generations, the population "evolves" toward an optimal solution. However, compared with other EAs, PBIL is simpler, robust and easy to implement. In addition, PBIL has fewer genetic operators than most of the EAs [16], [17].

In PBIL, the crossover operator is abstracted away and the role of the population is redefined [1], [3]. PBIL works with a probability vector (PV) instead of the whole population. The PV is used to control the random bit strings generated by PBIL and to create other individuals through learning. Learning in PBIL consists of using the current probability vector (PV) to create *N* individuals. The best individual is used to update the probability vector, increasing the probability of producing solutions similar to the current best individuals [3], [14]. It has been shown that PBIL outperforms standard GAs approaches on a variety of optimization problems including commonly used benchmark problems [1], [3], [12]-[13].

In [14]-[17], standard PBIL with single population and fixed learning rate was used to design power system controllers known as power system stabilizers (PSSs) for power systems. PBIL algorithm has given promising results in these papers.

The main purpose of a PSS is to damp low frequency oscillations arising from small and/or large disturbances in the power system. Without this device, a small disturbance may lead the system to instability and eventually collapse [14]-[18].

Recently, some authors have reported that PBIL suffers from diversity loss making the algorithm to converge to local optima [19]-[21]. The main reason for this is because of the fixed learning rate used in the standard PBIL. This fixed learning rate may not be as effective in dynamic environment to maintain the required trade-off between exploration and exploitation [22]-[24]. Also, the use of one probability vector (PV) to represent the whole population may reduce the diversity in the population and thereby degrade the global search ability of the algorithm [25].

Maintaining the diversity in PBIL's population is directly linked to the learning rate [19]. In [22], the authors investigated the effect of learning rate on the performance of PBIL. It was shown that using adaptive learning rate where the learning rate starts with a very small value and increases monotonically with the generation provides better results as compared to using a fixed learning rate. In [23], a hyper-learning scheme is proposed for PBIL in dynamic environment where the learning rate is temporarily raised whenever, the environment changes. In [24], opposition-based computing is used to alter the probability update rule with a more advanced mutation rule. However, the authors introduce new parameters which increase the complexity of the algorithm. In [25], adaptive updating of the learning rate was used where the learning rate varies according to the characteristics of specific search process. Maintaining the diversity in PBIL's population was achieved through the use of multiple probability vectors where every individual uses different probability vectors to generate its own children. In [26], multi-population PBIL is used to solve dynamic optimization problems. The ideas of parallel PBIL and dual PBIL are introduced. In parallel PBIL, the population is divided into two parts. Two probability vectors (PVs) are then used. One of these PVs is initialized to 0.5 and the other PV is randomly initialized. In Dual PBIL, the idea of complementary or duality is introduced and the PVs are initialized to complement each other

In this paper, the idea of parallel PBIL (PPBIL) using multi-population where both PVs are initialized to 0.5 is explored. The proposed approach is applied to a power system controller design. The effectiveness of the proposed algorithm is demonstrated by comparing it to the Adaptive PBIL (APBIL) introduced in [19], [22] and the standard PBIL (SPBIL) with fixed learning rate [14]-[17]. Simulation results show that the parallel PBIL (PPBIL) based on two-population is as effective as the Adaptive PBIL (APBIL) and performs better than the standard PBIL (SPBIL).

## 2 Overview of PBIL

PBIL is a technique that combines aspects of GAs and simple competitive learning derived from Artificial Neural Networks [1], [3], [12]-[13]. PBIL belongs to the family of Estimation of Distribution Algorithms (EDAs), which use the probability (or prototype) vector (PV) to generate sample solutions. There is no crossover operator in PBIL; instead a single probability vector is updated using solution with the highest fitness values [16]. Initially, the values of the probability vector are set to 0.5 to ensure that the probability of generating 0 or 1 is equal. As the search progresses, these values are moved away from 0.5, towards either 0.0 or 1.0.

Like in GA, mutation is also used in PBIL to maintain diversity. In this paper, the mutation is performed on the probability vector rather than on the sample solutions. A forgetting factor is used to relax the probability vector toward a neutral value of 0.5 [12], [14]-[17].

A summary of the PBIL used in this paper is given below [15]-[17]:

**Step 1.** Initialize element of the probability vector (PV) to 0.5 to ensure uniformly-random bit strings.

**Step 2.** Generate a population of uniformly-random bit strings and comparing it element-by-element with the PV. Wherever an element of the PV is greater than the corresponding random element, a "1' is generated, otherwise a '0' is generated.

**Step 3.** Interpret each bit string as a solution to the problem and evaluate its merit in order to identify the "Best".

**Step 4.** Adjust PV by slightly increasing PV (i) to favor the generation of bit strings which resemble "Best", if Best (i) = 1 and decrease PV(i) if Best(i) = 0.

**Step 5.** Apply the mutation and generate a new population reflecting the modified distribution. Stop if satisfactory solution is found. Otherwise, go to step 2.

The update rule of the probability vector is given as follows:

$$PV_i(g+1) = (1-LR)\times PV_i(g) + LR * B_i(g) \quad (1)$$

where 0<*LR*?1 is the learning rate that determines the distance the probability vector is pushed for each iteration, PV is the probability vector, *g* is the number of iterations or generations, *i* denotes each locus ($i$ = 1, 2. . . , $l$), and $l$ is the binary encoding length, *B* is the best solution.

Note that the mutation used for the PBIL adopted in this paper is slightly different from the one initially used in [1], [3]. The main idea in this paper is to relax the PV towards the neutral 0.5. The pseudocode for PBIL is shown in Fig. 1.

```
Begin
g:= 0;
//initialize probability vector
for $i$:=1 to $l$, do $PV_i^0 = 0.5$;
endfor;
while not termination condition do
generate sample S(g) from (PV(g) , pop.)
Evaluate samples S(g)
Select best solution B(g)
// update probability vector PV(g) toward best
solution according to (1)
//mutate PV(g)
Generate a set of new samples using the new
probability vector
g=g+1
```

**Figure 1**. Pseudocode for standard PBIL

It should be mentioned that the probability vector guides the search, which produces the next sample point from which learning takes place. The learning rate determines the speed at which the probability vector is shifted to resemble the best (fittest) solution vector. If the learning rate is fixed during the run (as it is the case with standard PBIL), it cannot provide the flexibility needed to achieve a trade-off between exploration and exploitation. Therefore, there is a possibility of premature convergence. Diversity in the population can be lost if the degree of exploitation is too high compared to the degree of exploration.

For the algorithm to perform optimally, it is crucial that diversity is not lost early in the run and that balance between exploration and exploitation is maintained as long as possible. Two methods that can be used to increase diversity will be discussed in the next section.

## 3 Overview of Diversity Increasing PBILs

### 3.1 Parallel PBIL

For the Parallel PBIL (PPBIL), two populations are used with two probability vectors ($PV_1$ and $PV_2$). Each probability vector is initialized to 0.5 and sampled to generate solutions independently from each other. The PVs are updated independently according to the best solution generated by each population. Initially, each probability vector has equal sample solutions. That is, the total population is divided by 2 and one half is assigned to each PV. As the run progresses, the probability vector (PV) that performs better is allowed to increase its share of samples. The sample sizes of the probability vectors are slightly adapted within the range [$\text{pop}_{min}$ $\text{pop}_{max}$] = [0.4*pop 0.6*pop] according to their relative performances. The probability that outperforms the other is increased by a constant value $\Delta$ = *LR**pop, where *LR* is the learning rate. For the PPBIL used in this paper, the learning rate was selected to be 0.1 for both probability vectors. Fig. 2 shows the pseudocode of PPBIL.

### 3.2 PBIL with Adaptive Learning Rate

The learning rate in the standard PBIL is usually fixed to a specific value. This means that the user has to spend a lot of time and try several values of the learning rate before deciding on the "best" value to use. If the value of the learning rate is too high, the algorithm will learn towards the best too quickly. This may lead to premature convergence and a lost of the diversity earlier in the run. In this case, the algorithm could converge to local optima. If on the other hand, the learning rate is too small, the algorithm may be slow to converge and will require more time to find the optimal solution. This may be computationally costly and a waste of resources. It is therefore crucial that the learning rate provides a trade-off between exploration and exploitation.

To develop the adaptive learning algorithm, we assume that at the start of the run, diversity will be needed for the algorithm to be able to explore thoroughly the search space. Therefore, at the start, a very small value of learning rate (LR$\approx$ 0) is selected [19], [22].

This means that at the beginning of the run, the emphasis is placed on the exploration rather than exploitation. As the run progresses and good individuals start to emerge, the emphasis is shifted gradually from exploration to exploitation. The learning rate is allowed to increase slowly and linearly according to the change in generation as given in the following equation:

$$LR(i) = LR * G(i)/G_{\max} \qquad (2)$$

where

*LR(i)* is the learning rate at the *ith* generation

*LR* is the final learning rate

*G(i)* is the *ith* generation

$G_{max}$ is the maximum generation allowed

```
Begin
g:= 0;
//initialize probability vector
for $i$:=1 to $l$, do $PV_{i1}^0 = PV_{i2}^0 = 0.5$;
endfor;
// initialize the sizes of the probability vectors
such that: pop1= pop 2= pop/2
while not termination condition do
generate sample $S_1(g)$ from ($PV_1(g)$ , pop1.)
generate sample $S_2$(g) from ($PV_2(g)$ , pop2.)
Evaluate samples ($S_1(g)$, $S_2(g)$)
Select best solutions $B_1(g)$and $B_2(g)$
// update probability vectors $PV_1$(g) and $PV_2(g)$
toward bests solution $B_1(g)$and $B_2(g)$ according
to (1)
If $f(B_1(g))> f(B_2(g))$
then  $pop_1$= min [($pop_1 + \Delta$)    $pop_{max}$]
If $f(B_1(g))< f(B_2(g))$
then  $pop_1$= max [($pop_1 - \Delta$)    $pop_{min}$]
$pop_2$= $pop-pop_1$
//mutate $PV_1(g)$ and $PV_2(g)$
```

**Figure 2**. Pseudocode of PPBIL

The pseudocode for APBIL is similar to that of standard PBIL except that the learning rate is now varying according to Eq. 2.

## 4 System Model and Operating Conditions

The power system considered in this paper is the two-area four-machine power system as shown in Fig. 3 [18], [27]. The system consists of two similar areas connected by a tie-line. Each area consists of two coupled conventional generator units, each generator is rated 900 MVA and 20 kV. The generators are modeled as round-rotor generators and represented by the detailed six order differential equations. The machines are equipped with simple exciter systems [14]. The dynamics of the system are described by a set of nonlinear differential equations. However, for the purpose of controller design, these equations are linearized around the nominal operating conditions as given below[15], [16]:

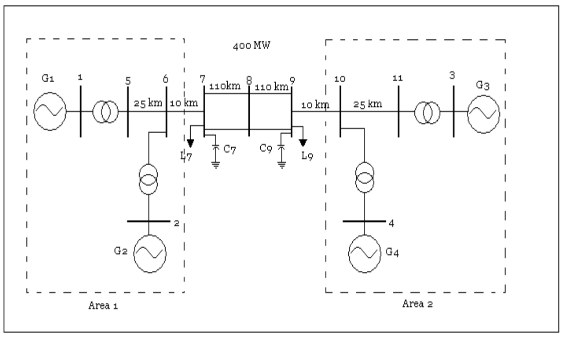

**Figure 3**. Power system model

$$\begin{aligned} \dot{x} &= A_o x + B_o u \\ y &= C_o x + D_o u \end{aligned} \qquad (3)$$

where

*x* are the state variables, *y* the system output and *u* the control input. $A_o$, $B_o$, $C_o$, $D_o$ are constant matrices of appropriate dimensions.

Three operating conditions were considered during the design of the controller. These are listed in Table 1. It is known that the system exhibits two local modes one in area 1 and the other in area 2 and one inter-area mode. However, for the purpose of this study only the inter-area modes will be considered since they are the most difficult to control.

Three cases are listed in Table 1. Case 1 is the light load condition, where about 200 MW of real power is transferred from area 1 to area 2. The system is stable for this case as can be seen by the negative value of the real part of the eigenvalue. Case 2 is the nominal condition, under this operating condition, there is a transfer of 400 MW power from area 1 to area 2. The system is unstable for this case, since the real part of the eigenvalue is positive. Case three is the heavy load condition where about 500 MW of power is transferred from area 1 to area 2.

Matlab Power System Toolbox (PST) was used for all the simulations [27].

**Table 1**. Operating Conditions

| **cases** | **Active Power Flow (MW)** | **Eigenvalues** |
|---|---|---|
| 1 | 200 | -0.35±j3.92 |
| 2 | 400 | 0.0096±j3.84 |
| 3 | 500 | 0.148±j3.09 |

## 5 Objective Function and Controller Design

### 5.1 Objective Function

As can be seen in Table 1, except for the light load condition, the system is unstable for the nominal and heavy operating conditions. This instability can be explained by the positive values of the real part of the eigenvalues. These oscillations should be damped for the system to perform adequately. The purpose of the controller design is to optimize the parameters of the power system stabilizers (iPSSs) simultaneously and in a coordinated and decentralized manner such that adequate damping is provided to the system for all the operating conditions considered in this paper, while keeping the structure of the PSS as simple as possible.

The objective function used is given in Eq. (4).

$$J = \max(\min(\zeta_{ij})) \tag{4}$$

$i = 1, 2, \ldots, n$ *eigenvalues*

$j = 1, 2, \ldots, m$ *operating conditions*

where $\zeta_{ij} = \frac{-\sigma_{ij}}{\sqrt{\sigma_{ij}^2+\omega_{ij}^2}}$ is the damping ratio of the $i^{th}$ closed – loop eigenvalue of the $j^{th}$ operating condition. $\sigma_{ij}$ is the real part of the eigenvalue and $\omega_{ij}$ is the frequency.

### 5.2 Controller Design

The structure of the widely used conventional PSS [18], [27] was adopted here. The structure of the controller to be designed is as shown in Eq. (5). It is required to simultaneously optimize the parameters of the controller such that adequate damping is provided for a wide range of operating conditions. In total, 10 parameters are to be optimized (i.e., 5 parameters for each area). The washout parameter $T_w$ is not critical and has not been optimized but was selected to be 10 s.

$$K(s) = K_p\left(\frac{sT_w}{1+sT_w}\right)\left(\frac{1+sT_1}{1+sT_2}\right)\left(\frac{1+sT_3}{1+sT_4}\right) \tag{5}$$

In Eq. (5), $T_1$to $T_4$ represent the time constants that need to be optimized to obtain adequate damping.

The parameter's configuration that was used in SPBIL is as follows:

Population: 10

Generation: 400

Learning rate: 0.1

Forgetting factor: 0.005

The parameter's configuration that was used in APBIL is as follows:

Population: 10

Generation: 400

Initial Learning rate: 0.0005

Final Learning rate: 0.2

Forgetting factor: 0.005

The parameter's configuration that was used in PPBIL is as follows:

Population: 10

Initial population for $PV_1$: 5

Initial population for $PV_2$: 5

Maximum population: 6

Minimum population: 4

Generation: 400

Learning rate: 0.1

Forgetting factor: 0.005

For all the controllers, the parameter domain is as follows:

$$0 \le K_p \le 30$$

$$0 \le T_1, T_3 \le 1$$

$$0.010 \le T_2,\ T_4 \le 0.3$$

## 6 Simulation Results

### 6.1 Fitness Values and Convergence Rate

In order to investigate the effectiveness of the PBIL algorithms, various aspects of the algorithms are compared such as best fitness values, mean fitness values and worst fitness values and the capability of the algorithm in maintaining the diversity in the population. For each algorithm, several independent runs were performed and the curves providing the best fitness values are selected and shown in Figs. 4-6.

**Table 2**. Best, MEAN and Worst Fitness Values

| Fitness | SPBIL | APBIL | PPBIL |
|---|---|---|---|
| Best | 0.484 | 0.514 | 0.502 |
| Mean | 0.434 | 0.383 | 0.434 |
| Worst | 0.166 | 0.090 | 0.124 |

Figs. 4-6 show the convergence rate of SPBIL, APBIL and PPBIL, respectively. It can be seen that APBIL and PPBIL converge to higher fitness values of 0.514 and 0.502, respectively, compared to 0.484 for SPBIL. However, APBIL converged to a slightly higher value of 0.5140 compare to 0.502 for PPBIL. From the simulation results, it can be seen that APBIL has more diversity in the population at the middle of the run between generation 100 and 200 than PPBIL and SPBIL. This can be attributed to the small value of learning rate, at these generations. Small learning rate increases the exploration of the algorithm and thereby introduces more diversity in the population at these generations. Unlike SPBIL which diversity is much more concentrated at the beginning of the run (i.e., between generation 1 and generation 150) , the diversity in PPBIL is somehow spread across all generations. For example, between generations 300 to 400, SPBIL and APBIL have converged (i.e., almost no diversity). On the other, PPBIL still has some diversity. Therefore, it can still explore the search space although at a limited pace.

**Table 3**. Number of Function Evaluations To Find The Best Solutions

| | |
|---|---|
| **SPBIL** | 3810 |
| **PPBIL** | 10610 |
| **APBIL** | 15950 |

Table 2 shows the comparison between the best, mean and worst, fitness values. It can be seen that on average SPBIL and PPBIL have practically the same fitness. The mean for PPBIL and SPBIL (approximately 0.434) is higher than that of APBIL (0.383). The main reason for this is that APBIL has much more spread, with the worst fitness value at 0.09 compared to 0.124 for PPBIL and 0.166 for SPBIL.

In terms of the distance between the best and the worst fitness values, APBIL has the highest distance (0.424), followed by PPBIL (0.378) and then SPBIL (0.318). This suggests that both APBIL and PPBIL have more diversity in their populations than SPBIL.

Table 3 shows the number of functions evaluations for each algorithm before the best fitness was found. It can be seen that SPBIL has the lowest function evaluations, APBIL has the highest function evaluations, and PPBIL is somehow in the middle. In terms of the speed in finding the best fitness value, SPBIL is better and APBIL is the worst. However, the best value found by SPBIL is lower than the best value found by APBIL and PPBIL. This suggests that although SPBIL converges faster, it converges to local optima, which may not be appropriate.

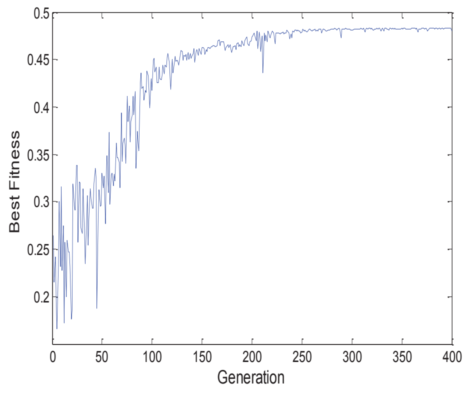

**Figure 4**. SPBIL convergence rate

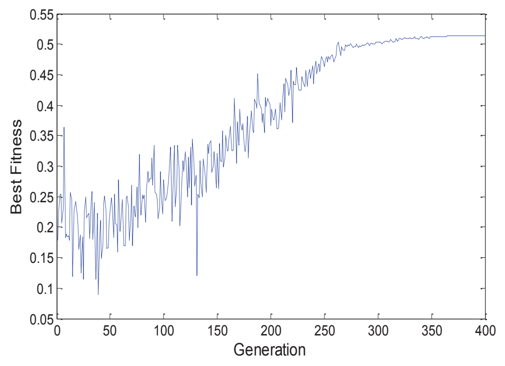

**Figure 5**. APBIL convergence rate

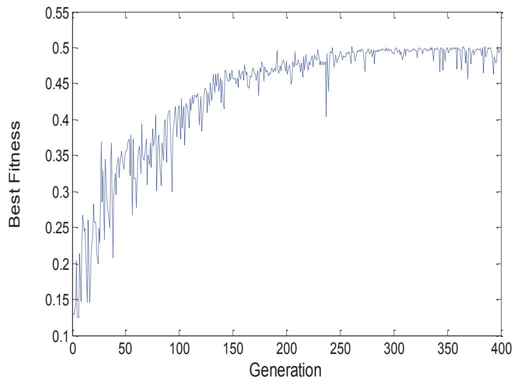

**Figure 6**. PPBIL convergence rate

## 6.2 Eigenvalue Analysis

Table 4 shows the eigenvalues and damping ratios in brackets of the closed-loop systems with the three controllers. Although the system has both local and inter-area modes, only the inter-area modes are considered because they are the most critical. It can be seen that PPBIL and APBIL give better performances (i.e., better damping ratios) than SPBIL. However, PPBIL provides slightly better damping than APBIL.

**Table 4**. Eigenvalues and Damping ratios

| **Case** | **SPBIL** $\lambda$ ($\zeta$) | **APBIL** $\lambda$ ($\zeta$) | **PPBIL** $\lambda$ ($\zeta$) |
|---|---|---|---|
| 1 | -1.13 ± j2.04 (0. 48) | -1.54 ± j2.57 (0.51) | -1.53 ± j2.53 (0.52) |
| 2 | -0.778 ± j1.78 (0.44) | -1.26 ± j2.11 (0. 51) | -1.26 ± j2.08 (0. 52) |
| 3 | -0.582 ± j1.25 (0.42) | -1.15 ± j1.52 (0.61) | -1.14 ± j1.54 (0.60) |

$\lambda$:eigenvalue, $\zeta$**:** damping ratio

## 7 Conclusion

Parallel PBIL based on multi-population that uses two probability vectors has been shown to increase the diversity in the population. This is important to prevent premature convergence that is inherent to the standard PBIL. The effectiveness of the proposed approach is assessed by comparing it to the Adaptive PBIL (APBIL) and the standard PBIL (SPBIL). It is shown that both the PPBIL and APBIL give better performances in terms of improving the damping of the system. They are able to maintain the diversity in the poppulation longer than the standard PBIL. In addition, they are able to maintaining the tradeoff between exploitation and exploration, with PPBIL slightly better in maintaining the diversity longer than APBIL.

## 8 Appendix

TABLE A. CONTROLLER PARAMETERS

| | | $K_p$ | $T_1$ | $T_2$ | $T_3$ | $T_4$ |
|---|---|---|---|---|---|---|
| **SPBIL** | Gen 1&2 | 29.99 | 0.993 | 0.010 | 0.056 | 0.300 |
| | Gen 3&4 | 29.63 | 0.128 | 0.035 | 0.127 | 0.162 |
| **APBIL** | Gen 1&2 | 29.97 | 0.946 | 0.300 | 0.063 | 0.010 |
| | Gen 3&4 | 29.88 | 0.439 | 0.010 | 0.053 | 0.300 |
| **PPBIL** | Gen 1&2 | 29.89 | 0.061 | 0.010 | 0.999 | 0.300 |
| | Gen 3&4 | 29.54 | 0.360 | 0.277 | 0.071 | 0.025 |

## 9 Acknowledgement

This work is based on the research supported in part by the National Research Foundation of South Africa, UID 83977 and UID 85503

## References

[1] S. Baluja, Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning, CMU-CS-94-163, Carnegie Mellon University, 1994.

[2] F. G. Lobo, and D.E. Golberg, The parameter-less genetic algorithm in practice, International Journal of Information Sciences 2004; 167, pp.217-32.

[3] S. Baluja, and R. Caruana, Removing the Genetics from the Standard Genetic Algorithm, Tech. Rep. CMU-CS-95-141), Carnegie Mellon University, 1995.

[4] J. H. Holland, Adaptation in nature and artificial systems. The University of Michigan Press, 1975.

[5] L. Davis, Handbook of genetic algorithms, International Thomson Computer Press, 1996.

[6] D. E. Goldberg, Genetic algorithms in search, optimization & machine learning. Addison-Wesley, 1989.

[7] K. Price, R.M. Storn, and J.A. Lampinen, Differential evolution: A practical approach to global optimization, Springer, ISBN 978-3-540-20950-8, 2005.

[8] T. Mulumba, and K. A. Folly, Power system stabilizer design: comparative analysis between differential evolution and population- based incremental learning", In: 20th Southern African Universities' Power Engineering Conference (SAUPEC ), 2011.

[9] M. Dorigo, and G Di Caro, The Ant Colony Optimization: a new teta-Heuristic, In: Evolutionary Computation (CEC), 1999.

[10] J. F. Kennedy, R. C. Eberhart R.C., & Y. Shi, Swarm Intelligence. Morgan Kaufmann, 2001.

[11] T. K. Das, and G.K. Venayagamoorthy "Design of Power System Stabilizers using Small Population Based PSO," IEEE PES General Meeting 2006.

[12] J. R. Greene, Population-Based Incremental Learning as a Simple,Versatile Tool for Engineering Optimization, In: Proceedings of the First International Conf. on EC and Applications, 1996, pp.258-269

[13] F. Southey, F. Karray, Approching Evolutionary Robotics through Population-Based Incremental Learning, In: IEEE International Conference on Systems, Man and Cybernetics, Vol. 2, 1999, pp. 710-715.

[14] KA Folly, Performance Evaluation of power system stabilizers based on Population-Based Incremental Learning (PBIL) Algorithm, International Journal of Power and Energy Systems, Vol. 33, Issue 7, 2011, pp. 1279-1287.

[15] K.A. Folly, Design of Power System Stabilizer: A Comparison Between Genetic Algorithms (GAs) and Population-Based Incremental Learning (PBIL), In: Proc. of the IEEE PES ,General Meeting, 2006

[16] K.A. Folly, Robust Controller Design Based on a Combination of Genetic Algorithms (GAs) and Competitive Learning, In: International Joint Conference on Neural Networks, 2007, pp. 3045-3050.

[17] P. Mitra, C. Yan, L. Grant, G.K. Venayagamoorthy, and K. Folly, Comparative Study of Population-Based Techniques for Power System Stabilizer Design, in Proc. of Intelligent System Applications to Power Systems, 2009.

[18] P. Kundur, Power System Stability and Control. McGraw – Hill, Inc. 1994.

[19] KA Folly, An Improved Population-Based Incremental Learning Algorithm, International Journal of Swarm Intelligence Research (IJSIR), Vol.4, No.1, 2013, pp. 35-61.

[20] C. Conzalez, J.A. Lozano and P. Larranaga, The convergence behavior of the PBIL Algorithm: A preliminary approach, In: Proc. of Artificial Neural Nets and Genetic Algorithms, 2001.

[21] R. Rastegar, A. Hariri, M. Mazoochi, The Population-Based Incremental Learning Algorithm Converges to Local Optima, Neurocomputing, 69, 2006, pp. 1772-1775.

[22] KA Folly, G.K. Venayagamoorthy, Effect of Learning Rate on the Performance of the Population-Based Incremental Learning Algorithm, In: Proc. of the International Joint Conf. on Neural Network (IJCNN), 2009.

[23] S. Yang and H. Richter, Hyper-Learning for Population-Based Incremental Learning in Dynamic Environment, In: IEEE Congress on Evolutionary Computation, 2009.

[24] M. Ventresca, H. R. Tizhoosh, A diversity Maintaining Population Based Incremental Learning Algorithm, Information Sciences, 178, 2008, pp. 4038-4056

[25] S. Y. Yang, S.L. Ho, G.Z. Ni, J.M. Machado and K.F. Wong, A new Implementation of Population-Based Incremental Learning Method for Optimizations in Electromagnetics, IEEE Trans. On Magnetics 43 (4), 2007, pp. 1601-1604.

[26] S. Yang and X. Yao, Experimental Study on Population-Based Incremental Learning Algorithms for Dynamic Optimization Problems, Soft Computing, 9(11), 2005, pp. 815-834

[27] G. Rogers, Power system oscillations, Kluwer academic Publishers, 2000.